

Paradigma objek

oleh Satrio Adi Rukmono

Pengertian paradigma objek

Di awal perkembangan komputer dan pemrograman, paradigma pemrograman yang populer adalah paradigma imperatif yang kemudian berkembang menjadi paradigma prosedural. Paradigma ini berangkat dari cara kerja komputer yang mengeksekusi urutan instruksi satu demi satu; pemrogram pun menulis program sebagai suatu urutan instruksi, langkah demi langkah, yang diperkaya dengan alur kendali (*control flow*) dalam bentuk percabangan dan perulangan, serta pemecahan program menjadi subprogram (fungsi dan prosedur). Salah satu kendala yang muncul dengan paradigma ini adalah skalabilitas. Dengan paradigma prosedural, program yang semakin besar cenderung memiliki semakin banyak *bug* sehingga memakan waktu untuk mencari dan membasminya. Paradigma objek diciptakan untuk menjawab permasalahan skalabilitas tersebut.

Istilah *object-oriented* dan paradigma objek dikemukakan oleh Dr. Alan Kay pada tahun 1960-an. Berlatar belakang ilmu biologi, Kay menyadari bahwa sel biologis dapat berkoordinasi secara masif pada skala trilyunan sel membentuk suatu sistem kompleks, yaitu organisme, yang memiliki kemampuan memperbaiki dirinya sendiri. Sebagai gambaran, diperkirakan sekitar 50 hingga 70 miliar sel dalam tubuh manusia mati setiap harinya, namun manusianya sendiri sebagai suatu organisme tetap hidup tanpa merasakan adanya perubahan. Bayangkan jika dalam perangkat lunak yang Anda buat terjadi *jutaan* error setiap harinya. Dapatkah Anda menjamin perangkat lunak tersebut tetap beroperasi dengan baik?

Dari latar belakang tersebut, konsep **objek** yang dicetuskan oleh Kay adalah semacam “sel” perangkat lunak yang dapat saling berkoordinasi dengan objek lainnya untuk membentuk suatu perangkat lunak yang utuh tanpa menyebabkan kegagalan perangkat lunak secara sistemik meskipun ada sebagian objek yang “mati”. Konsep ini oleh Kay dan timnya diimplementasikan dalam bahasa dan lingkungan pemrograman **Smalltalk** yang, meskipun saat ini tidak umum digunakan, menjadi pondasi bagi *graphical user interface* (GUI) dan *integrated development environment* (IDE) masa kini.

Aspek lain yang juga melatarbelakangi paradigma objek adalah *Maintainability*. Dengan ukuran dan kecepatan berubahnya perangkat lunak, *Maintainability* dapat

dianggap sebagai hal yang lebih penting dibandingkan seberapa cepat perangkat lunak dieksekusi. Membuat kode yang dapat dengan cepat dipahami oleh rekan satu tim dinilai lebih berharga untuk dapat menghasilkan produk yang memiliki lebih sedikit *bug* dalam waktu yang lebih singkat. Dalam paradigma objek, pemrogram berpikir dalam tingkatan abstraksi yang tinggi dan bukan pada detail implementasi struktur data, sehingga umumnya program yang dihasilkan mengorbankan sedikit kinerja namun lebih *maintainable*. Dengan demikian dapat disimpulkan bahwa paradigma objek lebih cocok digunakan untuk membuat perangkat lunak yang *high-level*, berskala besar, dan/atau berubah secara terus-menerus, dan sebaliknya kurang cocok untuk perangkat lunak *low-level* seperti *device driver* maupun untuk keperluan komputasi saintifik.

Ilustrasi

Paradigma prosedural

Seorang dosen berkata kepada mahasiswanya, “Berdiri, geser satu langkah ke kanan, jalan maju sampai ke depan kelas, hadap kanan, jalan maju sampai depan pintu, kalau pintunya tertutup, buka pintunya, maju dua langkah, dan tutup pintunya dari luar.”

Paradigma objek

Seorang dosen berkata kepada mahasiswanya, “Keluar.”

Dalam paradigma objek, sebuah objek bukan sekedar struktur penyimpanan data yang “dioperasikan”, melainkan suatu entitas pintar yang dapat menentukan langkah-langkah yang diperlukan untuk memenuhi suatu permintaan. Perhatikan bahwa pada ilustrasi di atas, dalam paradigma prosedural bukan saja dosen harus menjabarkan satu demi satu urutan langkah yang perlu dilakukan mahasiswa, ia juga harus mengetahui posisi duduk mahasiswa agar dapat dengan benar menginstruksikan “geser satu langkah ke kanan” dan bukan ke kiri. Sementara itu, pada paradigma objek, dosen menganggap mahasiswanya sebagai entitas yang pintar sehingga dengan memberikan pesan yang singkat saja, mahasiswa dapat mengetahui sendiri langkah-langkah yang harus ia lakukan untuk merespons pesan tersebut.

Dari berbagai latar belakang tersebut, Kay mendefinisikan paradigma objek sebagai “... only ***messaging***, ***local retention and protection*** and ***hiding of state-process*** [singkatnya kita sebut dengan ***isolation***], and ***extreme late-binding*** of all things.” (... hanya *messaging*, penyimpanan dan perlindungan lokal serta penyembunyian status dan proses, dan *extreme late-binding*.) Ketiga konsep ini akan dibahas lebih mendalam pada bab berikutnya.

Karakteristik pemrograman dengan paradigma objek

Dalam bukunya *Object-oriented Analysis and Design— with Applications*, Grady Booch (1994) mengatakan bahwa merancang program dengan pendekatan objek membutuhkan cara berpikir yang berbeda secara mendasar dibandingkan dengan pendekatan struktural. Pada pendekatan struktural, sudut pandang pemrogram adalah *how computers work*: bagaimana data direpresentasikan dalam memori dan urutan instruksi/proses terhadap data. Analisis dilakukan mengikuti pola “input-proses-output”. Pendekatan ini menarik inspirasi dari *proses bisnis* yang diterjemahkan ke struktur program, yaitu urutan instruksi ditambah alur kendali.

Sementara itu, dengan pendekatan objek sudut pandang pemrogram adalah objek apa saja yang terlibat dalam domain persoalan. Analisis dimulai dari identifikasi objek beserta perilakunya dan bagaimana objek-objek saling berinteraksi. Selain sel biologis, analogi lain yang menggambarkan paradigma objek dikemukakan oleh Brad Cox, pencipta bahasa **Objective-C**, dalam buku *Object-oriented programming: an evolutionary approach* (1986) yaitu *software-IC*, di mana sebuah objek diharapkan bisa bertindak sebagai komponen standar yang dapat dipertukarkan dengan objek lain yang “serupa” untuk konstruksi yang lebih kompleks, sebagaimana komponen mesin pada Revolusi Industri atau IC (*Integrated Circuit*) pada elektronika.

Beberapa karakteristik pemrograman berorientasi objek:

- Semua adalah objek, termasuk data yang dalam paradigma prosedural disebut sebagai bertipe primitif (integer, boolean, dsb.).
- Sebuah perangkat lunak merupakan koordinasi sekumpulan objek, yang masing-masing tersusun atas objek-objek lainnya.
- Objek itu “hidup” dan dinamis; perubahan terhadap objek dapat dilakukan bahkan saat perangkat lunak sedang berjalan.